Migrating to Carbon Events

Session 203

Migrating to Carbon Events

Xavier Legros
Mac OS X Evangelist
Apple Worldwide Developer Relations

Migrating to Carbon Events

Curt Rothert
Mac OS X Toolbox Engineer

What Is This Session About?

- Overview of Carbon Event model
- Migrating **WaitNextEvent** to Carbon Events

Who Is This Designed For?

- Add new features
- Migrate functionality to Carbon Events
- Good Mac OS X citizens!

What Benefits Will You Get By Migrating to Carbon Events?

- Allows modular design
- Rapid application development
- Compatible back to Mac OS 8.6 with **CarbonLib**
- Toolbox can add new standard features for free

What Benefits? (Cont.)

- Predictable behavior provided by Apple
- New features use Carbon Events
- Performance improvements . . .

Performance

- No polling
- No dead-end event processing
- Direct dispatching
- APIs specifically designed for performance
    - Go see Session 207: Performance with Carbon Events

Migrate at Your Own Pace

- Incremental changes
- Modular design

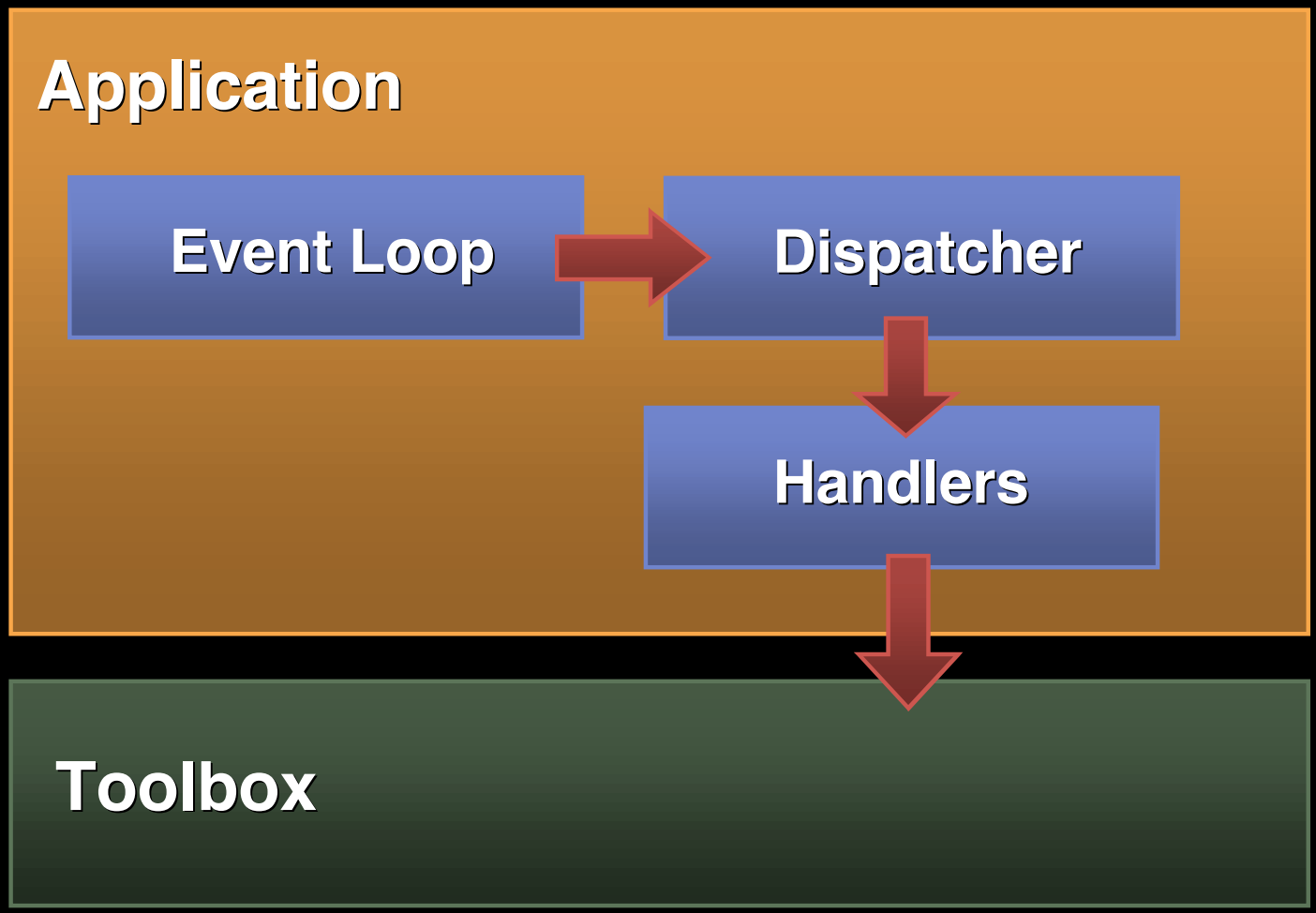
Where Appearance Manager defines look, Event Handling determines the feel

Carbon Events as a Framework

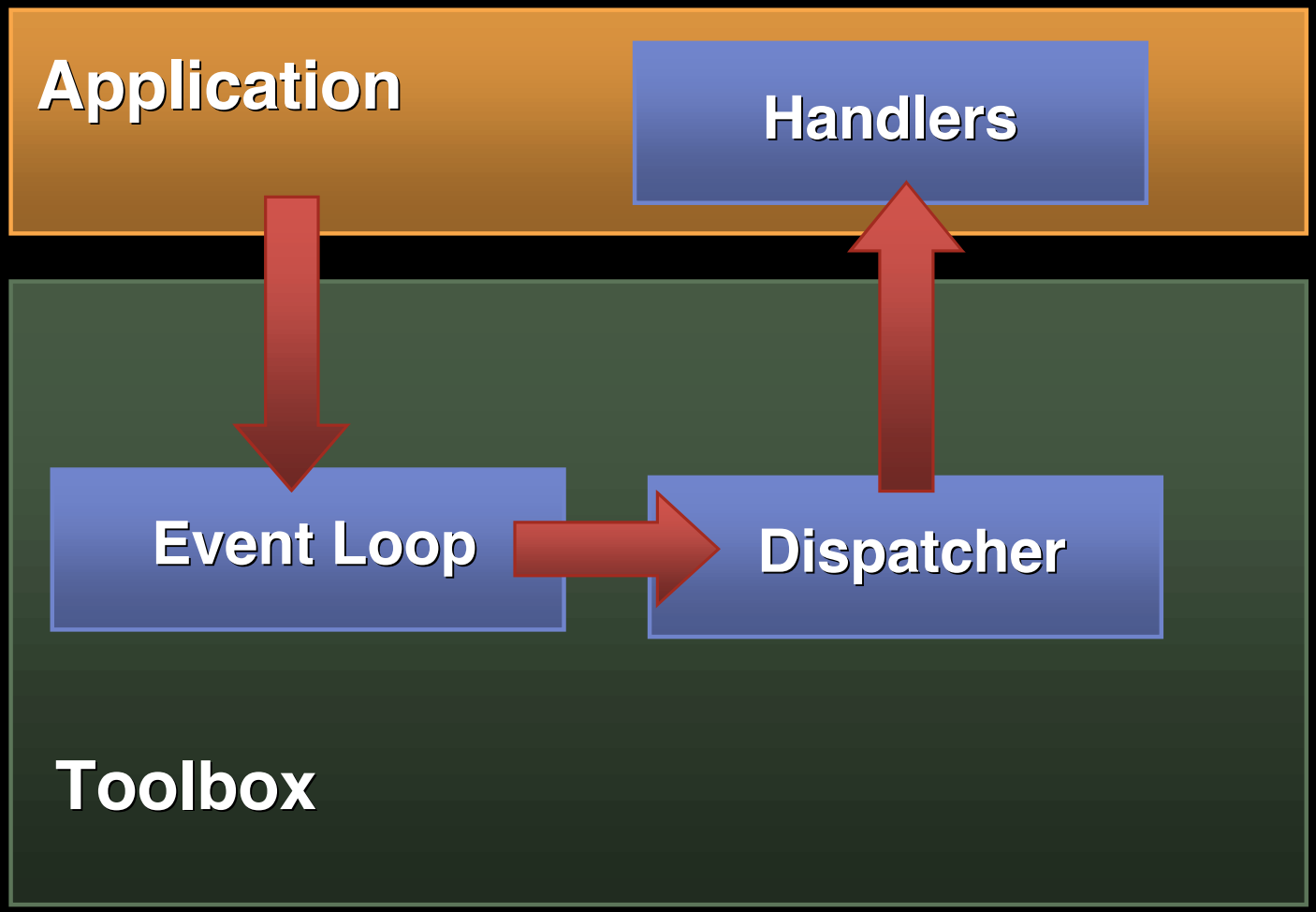
What Does This Mean for an Event-Driven App?

- With **WaitNextEvent**, you process each event and tell the toolbox what to do
    - **WaitNextEvent** layered on top of Carbon Events
- With Carbon Events, the toolbox processes each event and tells you what to do

WNE Model



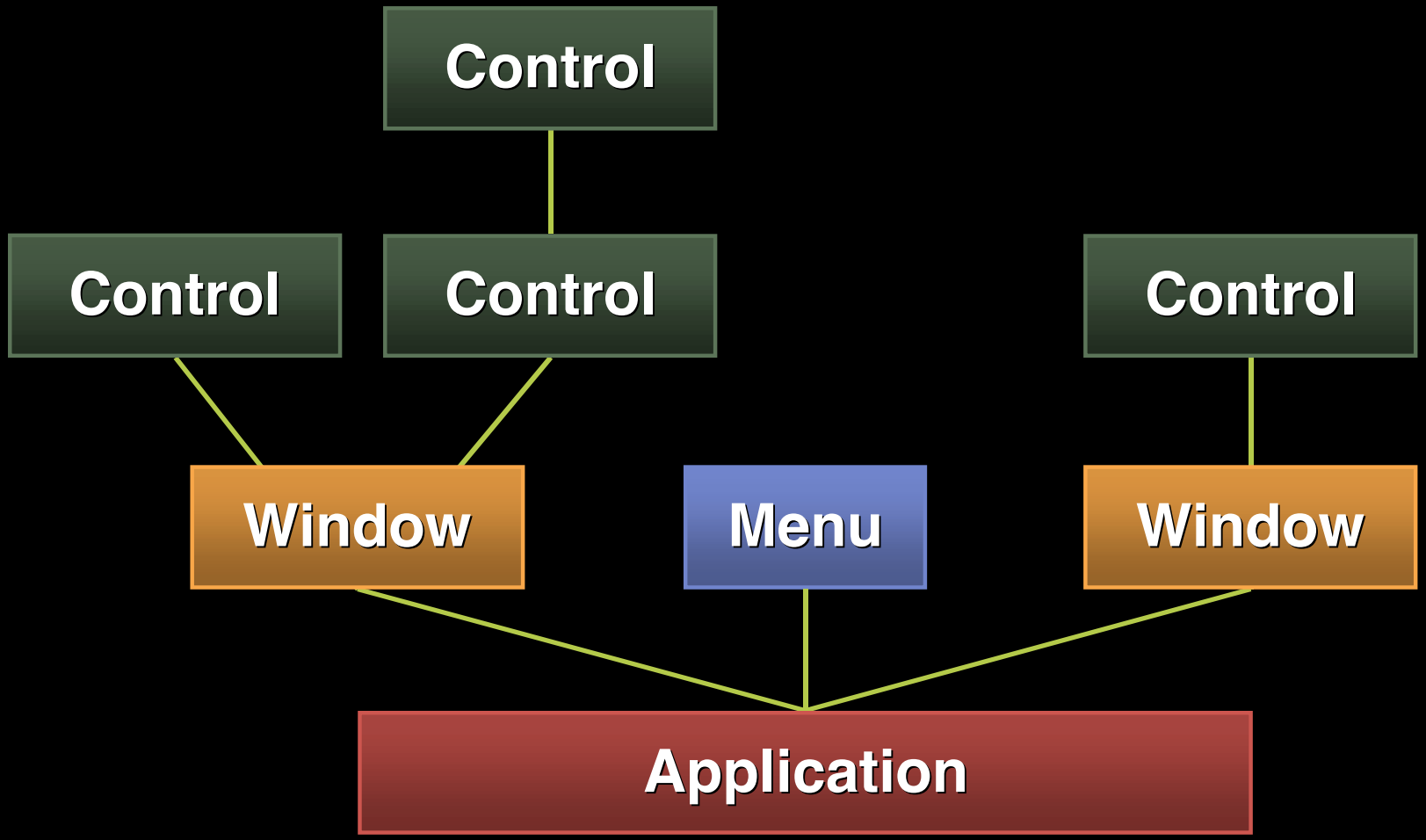
Carbon Event Model



Application Design

- Encourages event handling to be part of design
- Encourages code locality
- Ease of plugging in new modules
- Allows you to selectively override standard behaviors

Dispatching Hierarchy



Event Dispatching

- Hierarchical dispatching
- Standard handling
- Override standard behavior
- Modify standard behavior
    - Stackable handlers

CarbonEvent Data Types

- **EventTypeSpec**
  - To indicate what events you are interested in
- **EventRef**
  - The event notification type
- **EventTargetRef**
  - Where events are sent
- **EventHandlerRef**
  - A reference to your installed event handler

EventTypeSpec

```
struct EventTypeSpec {
  UInt32              eventClass;
  UInt32              eventKind;
};
```

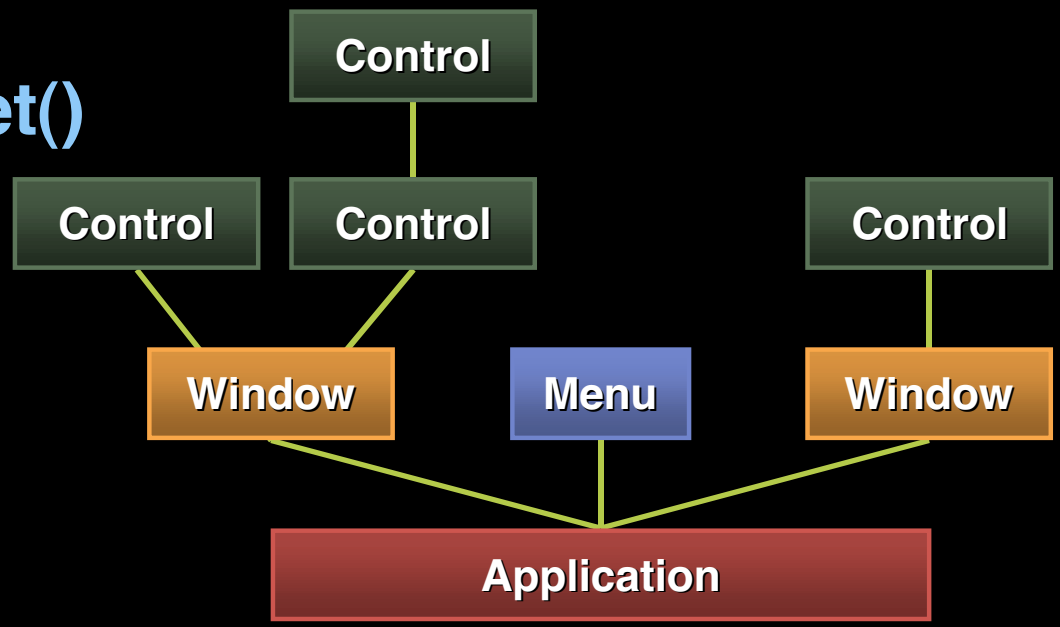
- Passed as an array

EventRef

- Can contain arbitrary data
    - Number of data types
    - Size
    - Type
- Data referenced by Parameter/Type pair

EventTargetRef

- Targets where your event handlers are installed and events get sent
- **GetFooEventTarget()**
  - Window
  - Control
  - Menu
  - Application
  - UserFocus
  - EventDispatcher



EventHandlerRef

- A reference to your event handler
- Returned with **InstallEventHandler**
- **AddEventTypesToHandler/ RemoveEventTypesFromHandler**

Carbon Event Notification

- Define your Carbon Event handler
- Install your handler on an Event Target

Event Handler Routines

```
OSStatus MyEventHandler(
        EventHandlerCallRef inHandlerCallRef,
        EventRef inEvent,
        void *inUserData);
```

- **inEvent** is the event that occurred
- **inUserData** is application defined
- Propagation is controlled by your return

Installing an Event Handler

- Use **InstallEventHandler**
- **NumTypes** can be determined with macro **GetEventTypeCount**
- Can remove handler with **RemoveEventHandler**

Demo

Bryan Prusha
CarbonEvent Dude

Steps for Migration

1. Replace tracking loops
2. Get rid of **nullEvent** idling with **WaitNexEvent**
3. Design new features with Carbon Events
4. Migrate event processing to use Carbon Events
5. Get rid of **WaitNextEvent** altogether

Step 1: Replace Tracking Loops

Migration Step

Change:

```
while( StillDown())
        GetMouse();
```

To:

```
err = TrackMouseLocation( inPort, &point, &result );
```

Advantages of TrackMouseLocation

- This blocks, freeing the CPU for other processes
- Note: Timers will fire

MouseTrackingRegion API

- New in Jaguar
- Entered/exited notification
- Can have many per window
- Cursor switching
- This is how window widgets work

Step 2: Get Rid of nullEvent Idle Processing

Problem: Idle Event Processing

- Processing at periodic times
  - Cursor blinking
  - Animation
- Processing when user is idle
  - Background sorting/searching

Problems Idling With WaitNextEvent

```
Boolean WaitNextEvent(
  EventMask      eventMask,
  EventRecord *  theEvent,
  UInt32         sleep,
  RgnHandle      mouseRgn);      /* can be NULL */
```

- Wastes CPU cycles
- Not modular
- Requires internal logic to maintain timers

Solution: EventLoopTimers

- Just install one timer per task
- Can use 3 classes of timers:
  - One-shot for singular events
  - Periodic for recurring events
  - Idle timers (in Jaguar)
- Session 207: Performance with Carbon Events

EventLoopTimer APIs

- **InstallEventLoopTimer**
- **SetEventLoopTimerNextFireTime**
- **RemoveEventLoopTimer**

Demo

Bryan Prusha
Idle King

Step 3: New Development With Carbon Events

- Become familiar with Carbon Events concepts
- Will not impact existing code

Developing With Carbon Events

- Application Features
  - Toolbars
  - Informational windows
  - Window definitions
- Become familiar with using Nibs

Adopting New Toolbox Features

- New features are Carbon Event only
  - Services Menu
  - MouseTrackingRegions
  - HIObject

Step 4: Migrate Event Processing

- You have two solutions
  - Use standard handlers
  - Override default behavior

Handling Events the Old Way

- Limited to 16 types of events
- Returns low-level event
- Handles idle processing
- Primitive mouse tracking
- You are required to do the proper dispatching
    - Boilerplate code that everyone has to do
    - Requires constant maintenance

Solution 1:
Use Standard Handlers

- You get standard behaviors for free!
- Future Toolbox features come for free!
- Migrate off of **WaitNextEvent**

Example: Windows

- To get the standard behavior: add **kWindowStandardHandlerAttribute** to your windows and you'll get this stuff for free
  - Resize
  - Drag
  - Tracking widgets, etc.

Solution 2: Customize Behavior

- Only install handlers for events that you are interested in customizing
- You can handle the high-level action, rather than the low-level event

Event Propagation Control

- Standard handling
  - Do nothing
- Override standard behavior
  - Install a handler for your event and return **noErr**
- Modify standard behavior
  - Preprocess
  - Postprocess: **CallNextEventHandler**
  - Both

Advanced Event Processing

- Create your own events and call **SendEventToEventTarget**
- Use Carbon Events for internal communication
  - Define your own **EventClasses**
  - Communicate between preemptive threads

Step 5: Remove WaitNextEvent

Steps:

- Follow the “Steps for Migration”
- Maximize sleep
- Replace **WaitNextEvent** with **RunApplicationEventLoop**
- Remove **WaitNextEvent** artifacts

RunApplicationEventLoop

- Enters the event loop
- Dispatches events to your handlers
- Returns only when **QuitApplicationEventLoop** is called

Demo

Bryan Prusha
Advanced CarbonEvent Developer

Tips:

- Event tracing:
  - **setenv EventTrace 1**
- Resources:
  - **http://developer.apple.com/carbon/tipsandtricks.html**

Summary:

- Migration is easy
- Migration is an incremental process
- Carbon Events are the future of the Toolbox
- Carbon Events adoption is a must have for being a good Mac OS X citizen

Documentation

Carbon Event Manager

- Carbon Porting Guide
- Handling Carbon Events
- Carbon Event Manager Reference

Documentation > Carbon > Operating System Services > Carbon Event Manager
developer.apple.com/techpubs/macosx/Carbon/oss/CarbonEventManager/carboneventmanager.html

Repeat After Me:

1. Replace tracking loops
2. Get rid of **WaitNextEvent** idling
3. Design new features with Carbon Events
4. Migrate event processing to use Carbon Events
5. Get rid of **WaitNextEvent** altogether

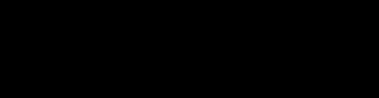
Roadmap

204 HIToolbox: An Architectural Overview	Hall 2 **Wed., 9:00am**
205 HIToolbox: Introducing HIView	Hall 2 **Wed., 10:30am**
206 HIToolbox: New Controls and Services	Hall 2 **Wed., 2:00pm**
207 Improving Performance With Carbon Events	Hall 2 **Wed., 3:30pm**

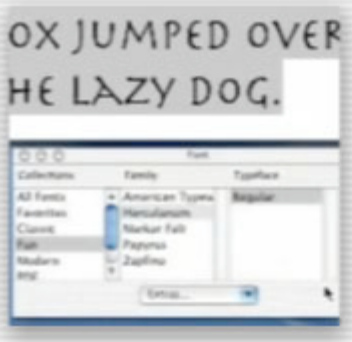
Who to Contact

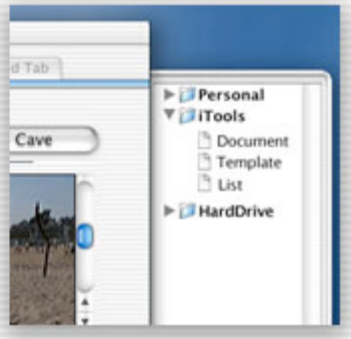
Xavier Legros
Mac OS X Evangelist
Apple Worldwide Developer Relations
xavier@apple.com

http://developer.apple.com/wwdc2002/urls.html



Q&A





Xavier Legros
Mac OS X Evangelist
xavier@apple.com

http://developer.apple.com/wwdc2002/urls.html


WWDC2002


WWDC2002


WWDC2002